This document is published in:

2011 Proceedings of the 14th International Conference on Information Fusion (FUSION 2011). Chicago, Illinois, USA 5-8 July 2011. IEEE, 2010, pp. 1771-1778

Neighborhood-based Regularization of Proposal Distribution for Improving Resampling Quality in Particle Filters

Enrique Martí, Jesús García and José Manuel Molina
Group of Applied Artificial Intelligence
University Carlos III of Madrid
Colmenarejo, Spain
Email: {emarti,jesus.garcia}@inf.uc3m.es, molina@ia.uc3m.es

***Abstract*—Particle Filter is a sequential Montecarlo algorithm extensively used for solving estimation problems with non-linear and non-Gaussian features. In spite of its relative simplicity, it is known to suffer some undesired effects that can spoil its performance. Among these problems we can account the one known as sample depletion. This paper reviews the different causes of sample depletion and the many solutions proposed in the existing literature. It also introduces a new strategy for particle resampling which relies in a local linearization of the proposal distribution. The particles drawn using the proposed method are not affected by sample impoverishment and can indirectly lead to better results thanks to a reduction in the plant noise employed, as well to increased performance because of requiring a lower number of particles to achieve same results.**

Keywords: Particle Filter, regularized, estimation, resampling, sample depletion, local linearization

I. Introduction to Particle Filters

Particle Filters (PFs) were born as an alternative to overcome the limitations of Kalman-like filters. Derived from Montecarlo simulation theory, they represent probability distributions using a set of weighted discrete samples —called particles—. The weight of a particle represents how likely it is to represent the unknown state of the system. This value is updated using measures, this is, imperfect evidences about the real state.

In theory, one of the main advantages of PFs is their simple implementation. It relies in simple simulation over a extense hypothesis set for substituting the matrices of Kalman-like filters —sometimes difficult to derive, and delicate to set properly—. However, experience says that achieving good results in concrete problems usually require specific solutions with a careful selection of techniques or even custom algorithms. Overcoming the natural limitations of PFs becomes a complex process that requires a good understanding of what is going on underneath.

One of the handicaps of PFs comes from the fact that the number of particles has to be finite and, hence, they have a limited capability for expressing the probability distribution of system state. This is more excruciating because the variance of the population tends to increase unbounded over time, and most of the particles will be in zones of the state space with negligible probability. As a consequence, most of the mass of the state probability distribution will be concentrated in a few (or just one) particles. This degrades the quality with which the population describes the state distribution and can cause the filter to diverge. However, this problem can be alleviated using a resampling algorithm.

The problem with basic resampling procedures is that they can fail to create new particles that actually describe the real state probability distribution. The special case where the resampled population collapses in a very reduced space receives the name of "sample depletion". This is caused because the particles recently created by resampling algorithm cannot be moved to fill the gaps between existing samples. This problem has been addressed since the early times of PF. Several solutions have been proposed, most of which try to find a continuous representation of the probility distribution function ($p.d.f.$). This approach receives the name of "regularization".

This paper is organized as follows. First of all, the resampling process will be briefly reviewed in section II. Following is section III, which analyzes some of the causes of sample depletion, and revisits the existing literature for solutions to this problem. After that, part IV will detail our proposal, including some remarks and considerations about its differences with previous solutions.

The paper ends with a battery of experiments in the context of Unmaned Aerial Vehicles (UAVs) navigation using an Inertial Measure Unit (accelerometer and gyroscope) and a GPS device. Section V gathers the equations for the simulation process and origin of the data. Finally, conclusions are shown and discused in VI.

II. Resampling in Particle Filters

This document assumes a knowledge of PF theoretical foundations. Readers expecting a thorough description of basic theory are encouraged to take a look at [1], [2] and the Technical Report [3] (do not confuse with the paper, also cited in this document)

Table I shows the pseudocode of Sampling Importance Resampling Particle Filter algorithm (SIR-PF) for reference.

Table I
PARTICLE FILTER PSEUDOCODE

$U_k \rightarrow$ *pdf* describing process noise

- Initialization
  - Draw N particles from initial state *pdf* $p(x_{k=0})$

$$p^i_{k=0} \sim p(x_{t=0}), \quad i = 1..N$$

  - Set weights to $w^i = 1/N$
- Repeat each time step:
  - Evolve particles using prediction model

$$p^i_k = f(p^i_{k-1})$$

  - Add sample from process noise to each particle

$$p^i_k = p^i_k + u^i, \quad u^i \sim U(x_k)$$

  - Update weights with last observation:

$$w^i_k = w^i_{k-1} \cdot p(y_k|x^i_k)$$

  - Normalize weights: $\tilde{w}^i_k = w^i_k / \sum_{a=1}^{N} w^a_k$
  - Estimate current state: $E[x_k] = \sum_{i=1}^{N} x^i_k \cdot \tilde{w}^i_k$
  - If needed, do resampling:
    1) Draw N new particles from previous population

$$\tilde{p}^i_k \sim \{p^i_k, \tilde{w}^i_k\}$$

    2) Add sample from process noise to each particle
    3) Reset the weights: $w = 1/N$

This section is focused on the last part, the resampling step. In the table, resampling follows the original proposal of Gordon [4], using the transition prior $p(x_k|x_{k-1})$ as proposal distribution for sampling new particles. This proposal is one of the most popular because, in spite that it does not take into account the last observation, the weight calculation is simplified to

$$w_k = w_{k-1} \cdot p(y_k|x_k) \tag{6}$$

More advanced proposals offer improved results, as $p(x_k|x_{k-1}, y_k)$ [5], but they are harder to implement. See [1] for a discussion on the topic.

Resampling has also be seen as moving samples towards zones of high interest in the probability distribution, instead of substituting them [6]. No matter the point of view, the ultimate goal of resampling is to arrange the population in

Sometimes, using a better proposal distribution requires additional work. This is the case of Auxiliary Particle Filters [7] (AuxPF), a technique that will be used in the experimental section. AuxPF work in the following way. When a resampling is required at time step $t = k$, particle indices are selected using a classical resampling algorithm. Instead of operating on the actual particles, the target will be a past state of the population at $t = k-n$ that was kept in memory. This past time particles are modified using a much reduced and controlled process noise, and after that are re-simulated up to current time instant. This method has two advantages: in first place, the process noise is less likely to create particles not fitting the expected posterior. Secondly, this proposal distribution achieves to integrate information about the last measure —something that is not true in the case of using transition prior as proposal. The question of how to select the indices among the old population so that distribution variance is kept has been subject of extensive research [5], [8], [9].

III. SAMPLE DEPLETION PROBLEM AND SMOOTHING TECHNIQUES

The population of a PF approximates a (rather continuous) probability distribution as a "set of deltas" surrounded by empty spaces. As the population of samples is assumed to be representative of the true probability distribution, the real density in such empty spaces is expected to be locally smooth, this is, more or less interpolable amongst nearby particles.

As shown in the introduction to PFs, typical resampling algorithms operate directly on the individual samples, because it is a simple and efficient process: samples are selected with probability proportional to their associated weight, cloned, and finally added a random noise sample according to the expected inaccuracy of the applied mathematical models (plant noise).

The random noise should introduce variability to fill the gaps between samples, but the approach has several drawbacks among which we can account:

- If plant noise is too small, resampling will not introduce the required variability in the resampled population. As a result, the particles will collapse in a few or just a single cluster, causing the filter to diverge in a few cycles. This is the case of the problem treated on this paper.
- The shape of the random noise (usually chosen to be Gaussian) can create particles whose distribution does not fit the one expected from the original ones. This is particularly bad in cases of low information availability. For instance, when very accurate sensor measures create sharply peaked distributions (a few particles retain most of the total weight, while most of the population approaches to zero density).
- The density of particles in the state space is not homogeneous, but the applied noise does not account for it, resulting in too small or too large perturbations depending on the case. As it will be described in next section, most regularization techniques based on local linearization also suffer this problem.

A. Literature review: Smoothing the proposal distribution

Existing literature provides a number of varied solutions to the resampling deficiency which leads to sample depletion. Some of them calculate a continuous analytical expression for the probability distribution described by particles, a procedure commonly termed as "regularization". The book [2] goes further and discerns between "regularization" and "local linearization" techniques, depending on how the analytical expression is obtained, as well as how is it used.

The group of regularization techniques feature algorithms as Kernel PFs [10], which approximate the cloud of particles through a weighted sum of kernels —usually Epanechnikov, or the computationally cheaper Gaussian function—, and use that continuous representation to generate the new population. The appropriate set of kernels can be found either by fitting

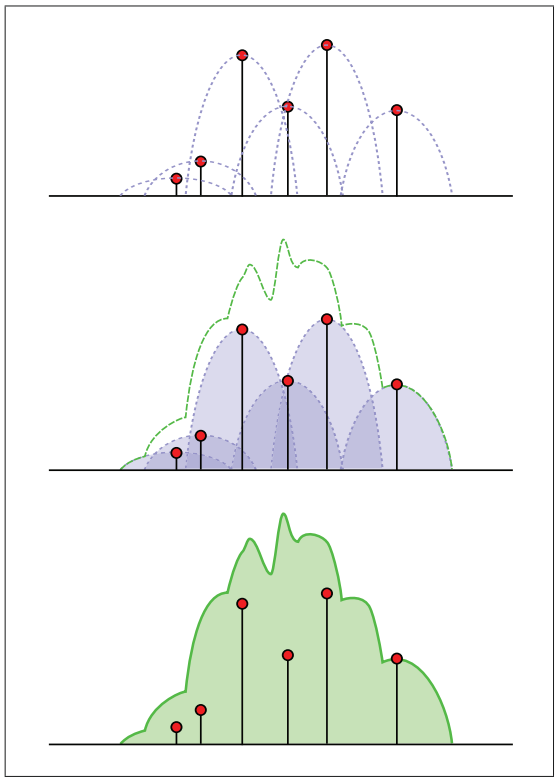

Figure 1. Generating a Regularized *pdf* from a set of discrete samples

them to the population, for example using an Expectation-Maximization (EM) algorithm. An alternative is attaching a kernel to each particle, which is used in [11] for object tracking in video. Figure 1 (based in the orginal found in [12]) illustrates this last procedure using Epanechnikov kernels. Once the set of kernels have been worked out, the new samples are generated. The "canonical" solution is to draw samples from the regularized probability distribution [13], although a popular alternative is to apply the mean-shift algorithm, which moves old particles following the gradient of the kernel approximation towards the mode of the distribution.

Local linearization techniques use a different approach: they typically attach a member of the Kalman Filter (KF) family to each sample. An example is the Unscented Particle Filter [14], which uses an UKF per particle that, in words of the authors, will propagate the sufficient statistics for each particle. The new population is resampled from the resulting sum of non-homogeneous kernels.

This concept was refined in [15] to reduce the computational burden of running N UKFs. First, the population (N particles) is expressed as a bank of G weighted UKFs. Additionally, both process and measure noises are simplified to a sum of P and M Gaussians. Assuming that $G \cdot P \cdot M \ll N$, its is computationally cheaper to perform prediction and measure update steps applying Kalman equations to all the $G \cdot P \cdot M$ combinations of Gaussian components, rather than sampling particles from them and applying the usual PF process. The only step which involves individual particles is resampling: N samples are drawn from the bank of UKFs, and then a new bank is fitted to the obtained population using a weighted expectation-maximization algorithm. A most recent work [16] combines a PF with a GMM for resampling stage without incuring in the heavy computational load of UKFs.

Another interesting question is how the probability distribution described by particles is regularized. Some of the techniques referenced above use population statistics to fit a mixture with a reduced number of components. Some others attach a kernel to each sample in the population. The first approach requires less computational power than the second, but it has the problem of selecting the appropriate number of components for the mixture. Moreover, if the regularized population is used for any part of the algorithm apart from resampling, there is the intrinsic risk of losing part of the properties of a pure PF algorithm. On the other hand, using a component per particle requires either defining the width of the kernel beforehand, or using a complete KF for each particle, which can increase the computational complexity of the algorithm to a cumbersome $\mathcal{O}(N^3)$. For the first option, there are some cases where the optimal size of the kernel has closed form [12]. Nonetheless, when the density of the particles in the state space is very uneven this appears to be a bad solution.

IV. Proposal: a new Neighborhood-Based Regularization strategy

This section presents a new algorithm called Neighborhood-Based Regularization (NBR), that has the same goal as other regularization techniques but does it from a different point of view. Instead of directly fitting a continuous probability distribution to the population, the resampled particles are generated according to the expected smoothed distribution, which is calculated in a case-by-case fashion using individual samples in a vicinity.

Our proposal relies in the usual resampling process of SIR-PF —this is, drawing N particles from the old population with probabilities proportional to their weights—. We assume that such a resampling strategy is correct in the sense that it preserves many "macroscopic" features of the original probability distribution, as the statistical moments. The NBR algorithm takes care of the existing gaps between neighbor particles. The samples obtained by applying a traditional resampling technique —stratified, systematic, residual resampling— are moved around in their surrounding space, so that they are more effectively distributed in the expected posterior state distribution. The rest of this section presents NBR algorithm in detail.

Let $P = \{p^i, w^i\}_{i=1..N}$ be a weighted population of N particles to be resampled. The first step consists in drawing N particles from P using a resampling algorithm. This results in the temporal set of particles $\tilde{P} = \{\tilde{p}\}_{i=1..N}$ where each sample is a copy of one from the original population $\tilde{p}^i = p^x, p^x \in P$. Particles in the final resampled population will

be those of the temporal population but randomly moved in the nearby probability space. The goal now is to calculate that space using the surrounding particles.

Given the i-th sample $\tilde{p}^i$ of the temporal population, the rest of the particles in the old population P are sorted from nearest to furthest. The employed metric is the Mahalanobis distance d_m, using the weighted covariance matrix Σ_P of the old population P (7). This approach is a fast way of making the algorithm invariant to the scaling of the different state dimensions. However, it can have some drawbacks in the case of distorted or extremely multimodal posteriors.

$$d_m(\tilde{p}^i, p^x) = \sqrt{(\tilde{p}^i - p^x)^T \cdot \Sigma_P^{-1} \cdot (\tilde{p}^i, p^x)} \tag{7}$$

The A nearest neighbors of $\tilde{p}^i$ are selected (8), and B of them are randomly chosen (9) with a probability proportional to their weights and distances (10). The values for A and B must hold $B \leq A \leq N$, and also $B \leq d$ where d is the dimensionality of the state space.

$$\begin{aligned} P_A = &\{p^a \in P\}_{a=1..A} \\ &d_m(\tilde{p}^i, p^a) < d_m(\tilde{p}^i, p^c) \quad \forall p^c \in (P - P_A) \end{aligned} \tag{8}$$

$$P_B = \{p^b \in P_A\}_{b=1..B} \tag{9}$$

$$Pr(p^b = p^a) \propto w^a \cdot d_m(\tilde{p}^i, p^a) \quad p^a \in P_A \tag{10}$$

For each of the selected neighbors $\{p_b \in P_B\}_{b=1..B}$, its difference vector with respect to the central particle p^i is calculated as $e_b^i = p^i - p_b$. These vectors are the basis of a certain subspace $\mathcal{S}_i$ of the state probability distribution around particle p^i —this is, considering the i-th particle the origin of coordinates: $\mathcal{S}_i = \{p^i, E_i = \{e_b^i\}\}$—. This can be seen in the lower part of figure 2 (assuming that the illustration is just the 2D representation of a higher-dimensional space).

A proper particle distributed following the target state can be obtained by sampling uniformly in the volume of the unit d-sphere in the subspace $\mathcal{S}$. Uniform samples in a d-sphere can be generated quite simply in two steps [17], [18]:

- Generate sample in the surface of the d-sphere.
  - Vector of d elements $D = [g_1, \dots g_d], g \sim \mathcal{G}(0; 1)$
  - Normalize elements $D = D / \parallel D \parallel$
- Select a radius for the previous sample
  - Take uniform number $u \sim U[0; 1]$
  - Modify radius according to dimensionality $r = u^{1/d}$
- Sample is $D \cdot r$

The first part is based in the fact that the space of multivariate gaussian vectors is radially simmetric (invariate against rotations) and, thus, their projections are uniform over the surface of the d-sphere. However, we are rather interested in generating the samples in the sector of the sphere delimited by the displacement vectors in E_i. This can be achieved by using a vector D with only positive values (which is no more than the absolute value of the usual vector $D = |D|$). As an additional note, the uniform number for the radius of the sample can be tuned to control the zones where particles can appear. In that case, the uniform number is generated using

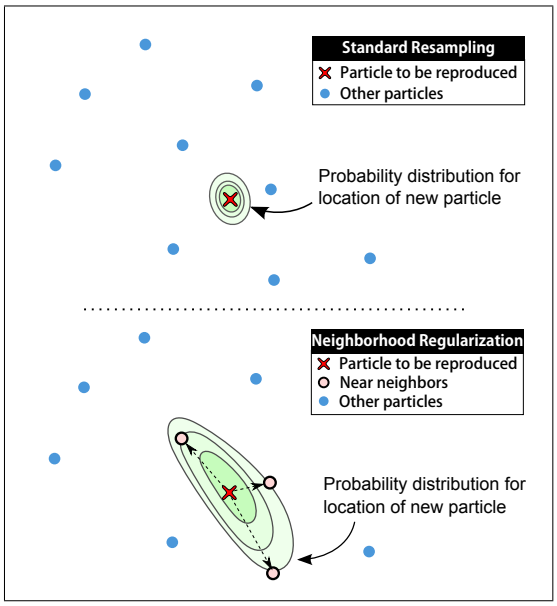

Figure 2. Neighborhood-based regularization avoids sample depletion

$u \sim U[ll; ul], \quad |ll|, |ul| <= 1$, where "$ll$" is the lower limit and "ul" is the upper bound for them.

The final i-th resampled particle is the one in the temporal population $\tilde{p}^i$, plus the obtained vector uniformly distributed on its neighborhood D. However, D has to be transformed from $\mathcal{S}$ coordinates to the canonical basis. This can be achieved by performing a basis change.

$$p^i = \tilde{p}^i + D \cdot E_i^b \tag{11}$$

Regarding the adecquate values for the involved parameters, the following facts have been observed:

- Parameter B is suggested to be a number below the dimensionality d of the state space, although reducing it too much causes the particles to be in a very reduced subspace of the target probability distribution. However, empirical results encourage the use of a not very large number of neighbors.
- The lower bound ll of the uniform distribution must be set to a small negative value in order to increase the variance of the resampled posterior. This is very important in the bounds of the population: when using only positive mixing values, particles in the convex hull do never generate new samples outside the original distribution, and hardly ever near the limit. In this case, the variance of the resampled population is always smaller than the original one, and can eventually cause a collapse of the estimated probability distribution.
- The only remark for ul is to set it to a positive value below one. The larger is B, the smaller the upper bound.

Just with illustrative purposes, in one of the performed experiments that used 500 particles to cover a 10-dimensional

space, good results are obtained setting $A \geq 50$, $2 \leq B \leq 5$, and $U[-0.1, 0.5]$.

The main advantage of the presented algorithm over existing approaches is its adaptability. The space between particles is smoothed using local information, so that zones with a high likelihood (that will usually be densely populated) will offer a finer detail. On the other hand, areas of low interest (described by just a few particles) will suffer a more aggresive smoothing. This approach deals right with uneven particle densities and multimodal distributions: if a particle selected for resampling is near a bound, their neighbors will tend to be its nearest ones, as shown in figure 2. This avoids introducing excessive variance in the population.

The main drawback of this technique is the increased computational load. While sampling a whole population has linear complexity —$\mathcal{O}(n)$, since it requires constant time for each particle—, the algorithm proposed in this paper is near-quadratic, given that generating a new particle from an original candidate involves searching its nearer neighbors. This cannot be reduced using improvements such as space partitioning techniques because of the high dimensionality of data. Take the example of a nearest neighbor search over a total of N points in k-dimensional space. According to [19] a KD-tree would be faster than exhaustive search only when $N \gg 2^k$, which is not the case of the problems solved using PFs.

However, even using the strictest formulation, the complexity of the proposed algorithm is $\mathcal{O}(n^2)$, which is less than the $\mathcal{O}(n^3)$ of some SPPF. In the proposed scenario of INS/GPS fusion, NBR-PF has lower computational requirements than techniques involving re-simulation as the Auxiliary PF. This is because the update step —and thus, resampling— has place when a GPS measure is received (about 1Hz), but prediction steps are much frequent —one per INS reading, between 20-100 Hz.

V. Experimental results

In the proposed experiments, a PF will be used to keep track of an UAV by using an Inertial Measure Unit (IMU) and a GPS sensor [20]. NBR will be compared with standard SIR-PF and an Auxiliary PF to demonstrate how it solves some existing problems and consistently achieves the best performance. The first part of this section presents the problem to solve along with its mathematical formulation, and the software tools designed for that task. Then, the results of the experiments will be analyzed.

A. Problem to solve - 6 DoF UAV flight

The dynamic model works over a vector containing position, speed and attitude of the UAV:

$$x[k] = \begin{bmatrix} p_n[k] & v_n[k] & q_n[k] \end{bmatrix} \tag{12}$$

Position and speed are expressed in Cartesian global coordinates, while the attitude is maintained in a quaternion. This vector can be extended to account for IMU biases just in case of trying to estimate them using the PF:

$$x_E[k] = \begin{bmatrix} x[k] & b_a[k] & b_\omega[k] \end{bmatrix} \tag{13}$$

IMU measures are interpreted as control inputs used in the prediction phase because describes how the state has to be propagated, i.e. how the state changes. The prediction equation is defined as: $x_n[k+1] = f(x_n[k], q(t), u(t), \Delta t)$ Following the convention of this document q(t) is the process noise, $u(t) = \begin{bmatrix} u_a(t) & u_\omega(t) \end{bmatrix}$ describes the control input composed by IMU measures from accelerometer and gyroscope, and Δt is the time elapsed since last prediction/update performed over the PF. It is important to remember that noises do not need to be Gaussian-distributed in a particle filter. Instead, let $\delta_q(t)$ $q(t)$ be a random sample drawn from (i.e. identically distributed to) the process noise. Typically, the noise is applied to second-order variables (this is, those that affect other variables during prediction), allowing it to propagate coherently to the whole state. The detailed functioning of the prediction function is presented now. It uses the above notation for a random sample drawn from process noise:

$$x_n[k+1] = f\left(x_n[k], q(t), u(t), \Delta t\right) = \begin{bmatrix} p_n[k+1] \\ v_n[k+1] \\ q_n[k+1] \\ a_n[k+1] \\ \omega_n[k+1] \end{bmatrix} \tag{14}$$

In first place, previous state $x_n[k]$ is modified by adding a random sample from plant noise $\delta x_n \sim q(t) : x'_n[k] = x_n[k] + \delta x_n$ Then components of prediction are calculated separately:

$$\begin{aligned} p_n[k+1] &= p'_n[k] + v'_n[k] \cdot \Delta t \\ v_n[k+1] &= v'_n[k] + \left(C_n^b[k] \cdot (u_a(t) - a'_n[k]) - g\right) \cdot \Delta t \\ q_n[k+1] &= -1/2 \cdot \Omega(t) \cdot q'_n[k] \cdot \Delta t \\ a_n[k+1] &= a'_n[k] = a_{n[k]} + \delta a \\ \omega_n[k+1] &= \omega'_n[k] = \omega_n[k] + \delta\omega \end{aligned} \tag{15}$$

Where:

$$\Omega[k] = \begin{bmatrix} 0 & u'_{\omega_x}[k] & u'_{\omega_y}[k] & u'_{\omega_z}[k] \\ -u'_{\omega_x}[k] & 0 & -u'_{\omega_z}[k] & u'_{\omega_y}[k] \\ -u'_{\omega_y}[k] & u'_{\omega_z}[k] & 0 & -u'_{\omega_x}[k] \\ -u'_{\omega_z}[k] & -u'_{\omega_y}[k] & u'_{\omega_x}[k] & 0 \end{bmatrix} \tag{16}$$

Is the matrix expressing attitude variation. It is created using the last bias-corrected gyroscope reading: $u'_\omega[k] = u_w(t) - \omega'_n[k]$ And $C_n^b[k]$ is the navigation-to-body rotation matrix, created from the quaternion attitude of state vector $\begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T = q'_n[k]$

B. Simulation model, data and execution environment

The motion of the UAV is described by the equation of a rigid body with six degrees of freedom (6DoF in advance), which determines the evolution of position, orientation, speed and accelerations suffered by a fixed-mass rigid body when it moves through a 3D space; this model also takes into account the inertia of the UAV.

We have developed a basic simulation process that generates realistic flight trajectories from an input consisting on forces and angular momenta in the body frame of the air vehicle.

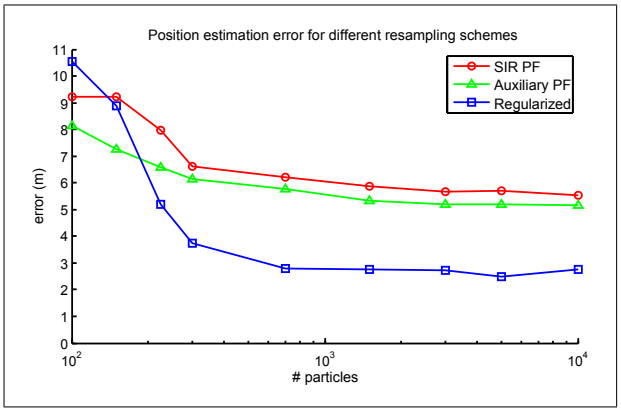

Figure 3. Position estimation error over number of particles for different resampling strategies

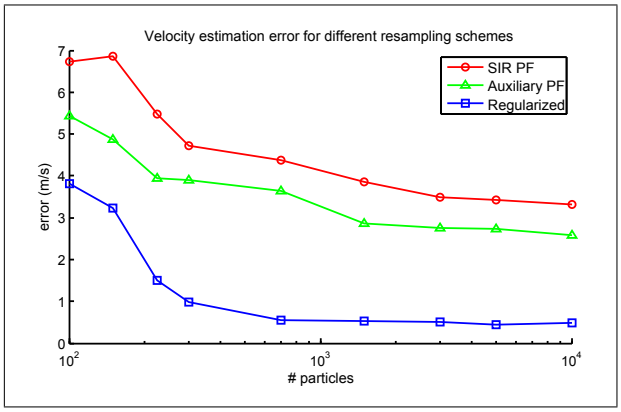

Figure 4. Velocity estimation error over number of particles for different resampling strategies

The simulator is base in MATLAB^{TM}Aerosim Aeronautical Simulation Block Set, which provides a complete set of tools for rapid development of detailed 6DoF nonlinear generic aerial vehicle models and also graphical view to check the behavior of system under test.

The simulator described above has been used to generate some of the most relevant situations in Air Traffic environments, such as straight flight at constant speed, straight flight with longitudinal acceleration and racetracks (a rectangle with two semicircles attached to its shorter sides, performed during the waiting time before landing in order to fit with the time scheduled). aiting. Additionally, some other synthetic trajectories have been created for pure testing purposes, such as 3-dimensional sequences of coordinated turns.

Sensors have been simulated using a simplified model. GPS sensor is considered to provide position as a tern (XYZ) indicating a point in a 3D orthogonal basis. Its error model is also very simple, just an additive zero-mean Gaussian noise with a certain variance, independently applied to each axis. For most of the experiments, this noise has selected to have $\sigma = 5m$, which gives a mean abslute positioning error of 8 meters.

The IMU is subject to a similar process, although our system supports also the adition of a bias and the typical cross-coupling effect that arises when the sensor axes are misaligned with respect to body frame.

VI. Results

The proposed implementation surpases standard SIR-PF and Auxiliary PF in every aspect. Not only obtains a better estimate for position, speed and attitude, but also needs a lower number of particles to approach its performance ceiling. Figures 3 and 4 compare the position and speed estimation error for the three algorithms for different population sizes. The results for each configuration are a 50-run average over a 2D random trajectory. The reason for the huge difference is that each filter is using the best possible configuration. This implies that, for SIR-PF and Auxiliary PF, the plant noise had to be increased nearly two orders of magnitude above its real value for avoiding degeneracy problems.

Table II
Time spent in resampling

	SIR-PF	Auxiliary PF	Regularized PF
Time (in seconds)	8.46 s	251 s	73 s
Part of total time	2.6%	43.8%	18.5%

The proposed resampling scheme works well with more than 300 particles. A very reasonable number taking into account that the state space has 10 dimensions. Needing less particles help reducing the expected computational requirements of the algorithm. Table II shows raw time and percentage of total execution time spent in resampling when filtering a 400 second long trajectory using a population of 1500 particles. We can see that the Regularized approach spends between 8 and 9 times more time in resampling than the SIR-PF, but just 3.5 times less than the Auxiliary PF. It is important to notice than the last entry in the table corresponds to half the resampling steps performed by the other two, but at the same time this means that the resampling quality of neighborhood-based regularized PF is clearly superior to SIR and Aux PFs. Furthermore, the experiments were performed with 1500 particles for the three options, but the RPF has been shown to require less than 500 for comparable accuracy levels. Because of this, we can say that in spite of its absolute computational cost, a PF can reduce computation time when switching to this resampling strategy.

NB-regularized PF does not only achieve better state estimations, but also converges faster than SIR-PF and AuxPF to stable values. Figure 5 shows how the NB-regularized reaches its peak accuracy around 50 seconds, while the other two resampling schemes continue reducing their estimation errors after 200 seconds. In the case of attitude estimation the differences are tighter, as observed in 6. Here, NBR-PF takes nearly 100 seconds to fall down to an average 0.5^{o} error —same time for AuxPF, in spite that this one is stabilized around 7^{o}—. The SIR-PF barely shows any improvement. A later experiment shown that, in this case, estimation accuracy is directly related

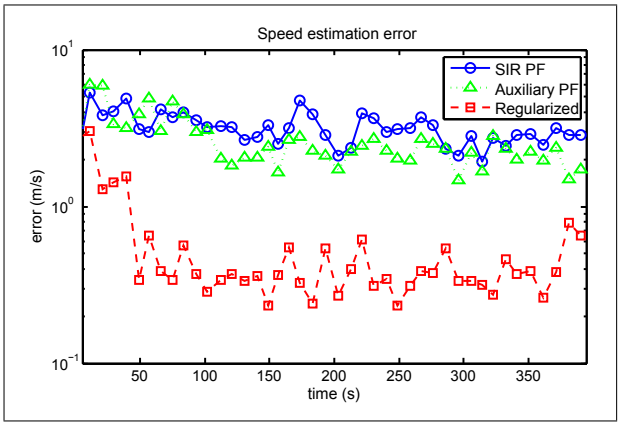

Figure 5. Evolution of velocity estimation error over time for the selected resampling strategies

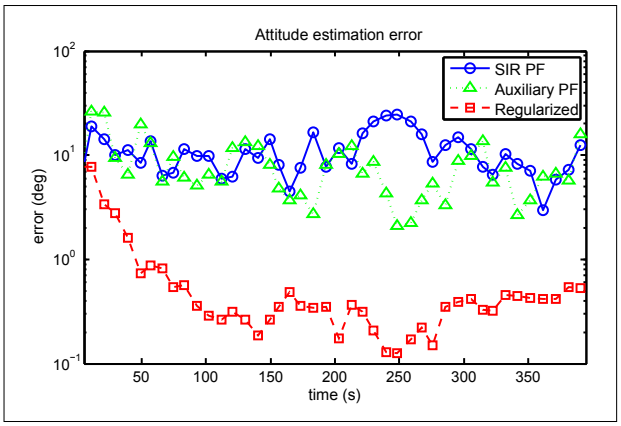

Figure 6. Evolution of attitude estimation error over time for the selected resampling strategies

with population variance. Thus, the convergence speed can be considered a measure of how fast a filter can reduce the uncertainty of initial state to a more realistic estimation. The aforementioned increased plant noise used for SIR-PF and AuxPF resampling makes impossible to obtain better values.

VII. Conclusions

This work has presented a new algorithm for regularizing the probability distribution suggested by the population of a Particle Filter. Applying this technique during the resampling step has demonstrated to avoid sample depletion, and to improve performance with respect to other versions as the Sampling Importance Resampling PF and Auxiliary Particle Filter.

In spite of lacking a direct comparison with other regularization techniques, the proposed algorithm appears to offer a good trade-off between performance and computational cost — especially when compared with alternatives as the Unscented PF, which requires a cubic amount of time with respect to the the population size—. It is even much cheaper than Auxiliary PF both in absolute and in relative terms. At the same time, neighborhood regularization can potentially capture a high number of moments of the underlying probability distribution, while not making any assumptions about it. One of its most important features is, probably, the adaptability: as it works directly with near particles, variable sample density does not affect it. Furthermore, it does not require defining parameters as the kernel width of other RPFs, which can even change along the process.

The proposed scheme has been tested and compared using the non-linear problem of aerial vehicle navigation with INS/GPS integration. This non-linear problem poses a challenge to Particle Filters because of its high dimensionality. However, the performed tests show that neighborhood-based regularization obtains speed estimation errors 4-6 times lower than those achieved by the other compared algorithms, with as few as 5 times less particles (around 300).

References

[1] S. Maskell, "Sequentially Structured Bayesian Solutions," Ph.D. dissertation, Cambridge University, 2004. [Online]. Available: http://www.simonmaskell.com/athesis.pdf

[2] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech Print on Demand, 2004. [Online]. Available: http://www.amazon.com/Beyond-Kalman-Filter-Particle-Applications/dp/158053631X

[3] R. V. D. Merwe, A. Doucet, N. de Freitas, and E. Wan, "The Unscented Particle Filter," Cambridge University Engineering Department, Tech. Rep., 2000. [Online]. Available: http://www.cs.ubc.ca/~nando/papers/upf.ps.gz

[4] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, no. 2, pp. 107–113, 1993. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=210672

[5] D. Crisan and A. Doucet, "A Survey of Convergence Results on Particle Filtering Methods for Practitioners," 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.6662

[6] R. Ansari, "Kernel particle filter for visual tracking," *IEEE Signal Processing Letters*, vol. 12, no. 3, pp. 242–245, Mar. 2005. [Online]. Available: http://www.citeulike.org/user/noxtoby/article/4193715

[7] M. K. Pitt and N. Shephard, "Filtering via Simulation: Auxiliary Particle Filters," *Journal of the American Statistical Association*, vol. 94, no. 446, p. 590, Jun. 1999. [Online]. Available: http://www.jstor.org/stable/2670179?origin=crossref

[8] J. S. Liu and R. Chen, "Sequential Monte Carlo Methods for Dynamic Systems," pp. 1032–1044, 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.1897

[9] G. Kitagawa, "Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models," *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, p. 1, Mar. 1996. [Online]. Available: http://www.citeulike.org/user/nimbot/article/4497411?citation_format=IEEEtran\#

[10] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 5, pp. 564 – 577, 2003. [Online]. Available: http://www.citeulike.org/user/changhai/article/1799558

[11] R. Ansari and A. Khokhar, *Multiple Object Tracking with Kernel Particle Filter*. IEEE. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1467318

[12] N. Oudjane and C. Musso, "Progressive correction for regularized particle filters," in *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=859873

[13] M. S. Arulampalam, S. Maskell, and N. Gordon, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," pp. 174–188, 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.117.1144

[14] R. van der Merwe, A. Doucet, N. D. Freitas, and E. Wan, "The Unscented Particle Filter," 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.9011

[15] R. V. D. Merwe and E. Wan, "Gaussian mixture sigma-point particle filters for sequential probabilistic inference in dynamic state-space models," *Acoustics, Speech, and Signal*, no. 3, pp. 4–7, 2003. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1201778

[16] S. Yang, D. Wen, J. Sun, and J. Ma, *Gaussian sum particle filter for spacecraft attitude estimation*. IEEE, Jul. 2010. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5555680

[17] M. E. Muller, "A note on a method for generating points uniformly on n-dimensional spheres," *Communications of the ACM*, vol. 2, no. 4, pp. 19–20, Apr. 1959. [Online]. Available: http://portal.acm.org/citation.cfm?id=377939.377946

[18] G. Fishman, *Monte Carlo*. Springer, 1996. [Online]. Available: http://www.amazon.com/Monte-Carlo-George-Fishman/dp/038794527X

[19] J. E. Goodman and J. O'Rourke, *Handbook of Discrete and Computational Geometry*, 2nd ed. Chapman and Hall/CRC, 2004. [Online]. Available: http://www.amazon.com/Handbook-Discrete-Computational-Mathematics-Applications/dp/1584883014

[20] F. Caron, E. Duflos, D. Pomorski, and P. Vanheeghe, "GPS/IMU data fusion using multisensor Kalman filtering: introduction of contextual aspects," *Information Fusion*, vol. 7, no. 2, pp. 221–230, Jun. 2006. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S156625350400065X